**Conv2D layer**

**Conv2D class**

**https://keras.io/api/layers/convolution\_layers/convolution2d/**

tf.keras.layers.Conv2D(

filters,

kernel\_size,

strides=(1, 1),

padding="valid",

data\_format=None,

dilation\_rate=(1, 1),

groups=1,

activation=None,

use\_bias=True,

kernel\_initializer="glorot\_uniform",

bias\_initializer="zeros",

kernel\_regularizer=None,

bias\_regularizer=None,

activity\_regularizer=None,

kernel\_constraint=None,

bias\_constraint=None,

\*\*kwargs

)

2D convolution layer (e.g. spatial convolution over images).

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use\_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

When using this layer as the first layer in a model, provide the keyword argument input\_shape (tuple of integers or None, does not include the sample axis), e.g. input\_shape=(128, 128, 3) for 128x128 RGB pictures in data\_format="channels\_last". You can use None when a dimension has variable size.

**Examples**

>>> # The inputs are 28x28 RGB images with `channels\_last` and the batch

>>> # size is 4.

>>> input\_shape = (4, 28, 28, 3)

>>> x = tf.random.normal(input\_shape)

>>> y = tf.keras.layers.Conv2D(

... 2, 3, activation='relu', input\_shape=input\_shape[1:])(x)

>>> print(y.shape)

(4, 26, 26, 2)

>>> # With `dilation\_rate` as 2.

>>> input\_shape = (4, 28, 28, 3)

>>> x = tf.random.normal(input\_shape)

>>> y = tf.keras.layers.Conv2D(

... 2, 3, activation='relu', dilation\_rate=2, input\_shape=input\_shape[1:])(x)

>>> print(y.shape)

(4, 24, 24, 2)

>>> # With `padding` as "same".

>>> input\_shape = (4, 28, 28, 3)

>>> x = tf.random.normal(input\_shape)

>>> y = tf.keras.layers.Conv2D(

... 2, 3, activation='relu', padding="same", input\_shape=input\_shape[1:])(x)

>>> print(y.shape)

(4, 28, 28, 2)

>>> # With extended batch shape [4, 7]:

>>> input\_shape = (4, 7, 28, 28, 3)

>>> x = tf.random.normal(input\_shape)

>>> y = tf.keras.layers.Conv2D(

... 2, 3, activation='relu', input\_shape=input\_shape[2:])(x)

>>> print(y.shape)

(4, 7, 26, 26, 2)

**Arguments**

* **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
* **kernel\_size**: An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
* **strides**: An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any dilation\_rate value != 1.
* **padding**: one of "valid" or "same" (case-insensitive). "valid" means no padding. "same" results in padding with zeros evenly to the left/right or up/down of the input.
* When padding="same" and strides=1, the output has the same size as the input.
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch\_size, height, width, channels) while channels\_first corresponds to inputs with shape (batch\_size, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be channels\_last.
* **dilation\_rate**: an integer or tuple/list of 2 integers, specifying the dilation rate to use for dilated convolution. Can be a single integer to specify the same value for all spatial dimensions. Currently, specifying any dilation\_rate value != 1 is incompatible with specifying any stride value != 1.
* **groups**: A positive integer specifying the number of groups in which the input is split along the channel axis. Each group is convolved separately with filters / groups filters. The output is the concatenation of all the groups results along the channel axis. Input channels and filters must both be divisible by groups.
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (see keras.activations).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix (see keras.initializers). Defaults to 'glorot\_uniform'.
* **bias\_initializer**: Initializer for the bias vector (see keras.initializers). Defaults to 'zeros'.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix (see keras.regularizers).
* **bias\_regularizer**: Regularizer function applied to the bias vector (see keras.regularizers).
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation") (see keras.regularizers).
* **kernel\_constraint**: Constraint function applied to the kernel matrix (see keras.constraints).
* **bias\_constraint**: Constraint function applied to the bias vector (see keras.constraints).

**Input shape**

4+D tensor with shape: batch\_shape + (channels, rows, cols) if data\_format='channels\_first' or 4+D tensor with shape: batch\_shape + (rows, cols, channels) if data\_format='channels\_last'.

**Output shape**

4+D tensor with shape: batch\_shape + (filters, new\_rows, new\_cols) if data\_format='channels\_first' or 4+D tensor with shape: batch\_shape + (new\_rows, new\_cols, filters) if data\_format='channels\_last'. rows and cols values might have changed due to padding.

**Returns**

A tensor of rank 4+ representing activation(conv2d(inputs, kernel) + bias).

**Raises**

* **ValueError**: if padding is "causal".
* **ValueError**: when both strides > 1 and dilation\_rate > 1.